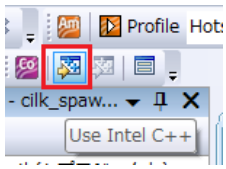
インテル Parallel Composerの新機能――並列プログラムを容易に実装できる「インテル Cilk Plus」入門



（http://sourceforge.jp/projects/hpc-parallel/wiki/%E3%82%A4%E3%83%B3%E3%83%86%E3%83%AB_Parallel_Composer%E3%81%AE%E6%96%B0%E6%A9%9F%E8%83%BD%E2%80%95%E2%80%95%E4%B8%A6%E5%88%97%E3%83%97%E3%83%AD%E3%82%B0%E3%83%A9%E3%83%A0%E3%82%92%E5%AE%B9%E6%98%93%E3%81%AB%E5%AE%9F%E8%A3%85%E3%81%A7%E3%81%8D%E3%82%8B%E3%80%8C%E3%82%A4%E3%83%B3%E3%83%86%E3%83%AB_Cilk_Plus%E3%80%8D%E5%85%A5%E9%96%80_p1/attach/icon.png）

並列プログラミング向けのコンパイラやデバッガ、各種ライブラリを提供するインテル Parallel Compsoserには、並列プログラミング向けの言語拡張「インテル Cilk Plus」が含まれている。これを利用することで既存のプログラムを容易に並列化したり、より簡潔にアルゴリズムを記述できる。本記事では、このCilk Plusについて機能や使い方を説明する。

Parallel Composerの強力な新機能「インテル Cilk Plus」

インテル Parallel Studio 2011にはさまざまな新機能が搭載されているが、そのなかでも注目したいのがインテル Parallel Composerに含まれるインテル Cilk Plusだ。Cilk Plusは、次のような特徴を持つ言語拡張だ。

- ループや連続した処理をキーワード指定だけで容易に並列化できる（cilk_spawn/cilk_sync/cilk_forキーワード）
- 配列に対する処理の記法が拡張され、「[:]」という表記で複数の配列要素に渡る処理を容易に記述できる
- 配列を引数に取る関数を容易に並列化できる（elemental function）
- 最小限のオーバーヘッドで排他制御を行える（reducer）
- 実装時に生成するスレッド数を考慮する必要がなく、容易にスケーラブルなプログラムを記述できる
- 覚えるべきキーワードや機能が少ないため習得が容易

たとえば**リスト1**はクイックソートをシンプルに実装したものだが、これをCilk Plusで並列化する場合、後半の「my_qsort」関数を再帰呼び出ししている部分に「cilk_spawn」キーワードを付け、最後に「cilk_sync」キーワードを追加するだけで並列化が完了する（**リスト2**）。

リスト1 クイックソートの例

```
void my_qsort(int* begin, int* end) {
 int* left = begin;
 int* right = end;
 int n;

 while (1) {
  while (*left  *begin)
   left++;
  while (*begin  *right)
   right--;

  if (left = right)
   break;

  n = *left;
  *left = *right;
  *right = n;
  left++;
  right--;
 }

 if (begin  left-1)
  my_qsort(begin, left-1);
 if (right+1  end)
  my_qsort(right+1, end);
}
```

リスト2 リスト1をCILK PLUSを用いて並列化する

```
 if (begin  left-1)
  cilk_spawn my_qsort(begin, left-1);    ←「cilk_spawn」を追加
 if (right+1  end)
  my_qsort(right+1, end);
 cilk_sync;    ←「cilk_sync;」を追加
```

ごく簡単な修正であるが、筆者の環境（Core 2 Duo E6550/2.33GHz・2コア、メモリ2GB）で1億個の要素を持つint型配列をソートするのに必要な時間を計測・比較したところ、これだけで3割ほどの実行時間短縮が確認できた（**表1**）。

表1 CILK PLUSによる並列化の結果（ソート実行時間比較）

並列化前	並列化後
10611ミリ秒	7662ミリ秒

また、配列に対する演算を次のように簡潔に記述することも可能となる。

```
double A[6];
double B[] = { 1.0, 4.2, 4.2, 5.3, 7.4, 3.2 };
double C[] = { 3.1, 8.9, 1.4, 2.2, 1.0, 2.4 };
double d;

A[:] = 0.0;     ←配列Aの要素をすべて0.0に初期化
d = __sec_reduce_add(A[:] * B[:]);     ←BとCの内積を計算
A[:] = B[:] + C[:];     ←A[0] = B[0] + C[0]、 A[1] = B[1] + C[1]、……に相当
```

Cilk Plusはこのように非常に簡単に導入でき、また並列化によるパフォーマンスの向上だけでなくコードをより簡潔に記述できるというメリットがある。以下ではこのCilk Plusについて、基本的な機能や文法などのポイントを紹介していく。

Cilk Plusの歴史

Cilk Plusは元々はMITの「The Cilk Project（http://supertech.csail.mit.edu/cilk/）」により、ANSI Cベースの並列プログラミング言語「Cilk」として開発されていた。Cilkは当初High Performance Computing（HPC）と呼ばれる、スーパーコンピュータなどの高性能コンピュータ向けに開発されていたが、Leisersonn教授らが起こしたベンチャー企業Cilk Artsが商用化、C++対応やさまざまな新機能を追加した「Cilk++」としてリリースされた。その後Cilk ArtsはIntelに買収され、仕様変更とともに名称もCilk Plusに変更、Parallel Studioの一部となった。

なお、CilkはGCCをコンパイラとして利用しており、現在でもGPLでリリースされている。CilkとCilk Plusではspawn/syncキーワードなど共通している部分が多いため、Cilkを利用したことのあるユーザーであればCilk Plusも容易に扱えるだろう。ただし、C++対応やarray extensionsなどClik PlusにはCilkには含まれていない独自の拡張機能も多くあるので、ドキュメントなどで詳細を確認していただきたい。